WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, September 19, 2005

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L18	L17 and ((integrated circuit) or chip or (programmable processor))	4
☐	L17	L16 and l12	6
☐	L16	19990507	4229
☐	L15	((receive or receiving or input) adj2 processor) same ((transmit or transmitting or output) adj2 processor)	6784
☐	L14	L13 and (packet near5 (switch or switching))	5
☐	L13	L12 and ((integrated circuit) or chip or (programmable processor))	57
☐	L12	19990507	171
☐	L11	context processor	531
☐	L10	stream context processor	0
☐	L9	19990507	5
☐	L8	L7 and context	0
☐	L7	((integrated circuit) or chip or (programmable processor)) same (packet near5 s(witch or switching))	12
☐	L6	19990507	4
☐	L5	19990507	1
☐	L4	L3 and chip	6
☐	L3	processor near8 stream near8 aggregate	14
☐	L2	(plurality adj2 processor) near8 (pluralty adj2 stream)	0
☐	L1	(plurality adj2 processor) near8 (pluralty adj2 stream) near8 (aggregator or aggregate or coordinate or coordinating or combine or combining)	0

END OF SEARCH HISTORY

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L9: Entry 1 of 5 File: USPT Jul 1, 2003

DOCUMENT-IDENTIFIER: US 6587452 B1
TITLE: High performance signal structure with multiple modulation formats

Application Filing Date (1):
19990104

Detailed Description Paragraph Table (2):
TABLE 2 10 Mb/s Packet Switching Specifications Equipment Base Terminal Operating Band 2400-2480 MHz 2400-2480 MHz Bandwidth 70 MHz 70 MHz RF Bandwidth 70 MHz 70 MHz Duplex Method packet switched packet switched Multiple Access GBT-CDMA GBT-CDMA Technique Number of Transmitter 2.sup.31 - 1 2.sup.31 - 1 Chip-Sequences TX data Rate: Traffic 9.6 Mb/s 9.6 Mb/s Signaling/APC Control Frame Length variable variable Data Modulation BPSK BPSK Spreading Technique Direct Sequence Direct Sequence Sequence Length Header 48 chips 48 chips Data 16 chips 16 chips Chip Rate 38.4 Mchips/s 38.4 Mchips/s Processing Gain 12 dB 12 dB Transmitter power 100 mW 100 mW (max) Device Range (free 0.4 km 0.4 km space) Number of Antenna omni omni Sectors Capacity 2 simultaneous users

Detailed Description Paragraph Table (4):
TABLE 4 10 Mb/s Packet Switching Specification Equipment Base Terminal Operating Band 2400-2480 MHz 2400-2480 MHz RF Bandwidth 26 MHz 26 MHz Duplex Method Time Division Duplex Time Division Duplex Multiple Access GBT-CDMA GBT-CDMA Technique Number of Transmitter 2.sup.31 - 1 Chip-Sequences TX Data Rate: Traffic 384, 144, 128, 64, 32 384, 144, 128, 64, 32 Signaling/APC Kb/s Kb/s Forward Error Coding Rate-1/2 Constraint Rate-1/2 Constraint Length-7 Length-7 Convolutional Code Convolutional Code Interleaver 5 ms 5 ms Control Frame Length 500 .mu.sec 500 .mu.sec Data Modulation BPSK BPSK Spreading Technique Direct Sequence Direct Sequence Sequence Length 6,930,000 chips 6,930,000 chips Chip Rate 38.4 Mchips/s 38.4 Mchips/s Processing Gain 12 dB 12 dB Transmitter power 100 mW 100 mW (max) Service Range (free 0.8 km 0.8 km space) Number of Antenna omni omni Capacity 2 simultaneous users

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L9: Entry 2 of 5 File: USPT Jul 17, 2001

DOCUMENT-IDENTIFIER: US 6262971 B1
TITLE: Multichannel spread-spectrum packet

Application Filing Date (1):
19981030

Detailed Description Paragraph Table (2):
TABLE 2 10 Mb/s Packet Switching Specifications Equipment Base Terminal Operating Band 2400-2480 MHz 2400-2480 MHz Bandwidth 70 MHz 70 MHz RF Bandwidth 70 MHz 70 MHz Duplex Method packet switched packet switched Multiple Access GBT-CDMA GBT-CDMA Technique Number of Transmitter 2.sup.31 -1 2.sup.31 -1 Chip-Sequences TX data Rate: Traffic 9.6 Mb/s 9.6 Mb/s Signalling/APC Control Frame Length variable variable Data Modulation BPSK BPSK Spreading Technique Direct Sequence Direct Sequence Sequence Length Header 48 chips 48 chips Data 16 chips 16 chips Chip Rate 38.4 Mchips/s 38.4 Mchips/s Processing Gain 12 dB 12 dB Transmitter power 100 mW 100 mW (max) Service Range (free 0.4 km 0.4 km space) Number of Antenna omni omni Sectors Capacity 2 simultaneous users

Detailed Description Paragraph Table (4):
TABLE 4 10 Mb/s Packet Switching Specification Equipment Base Terminal Operating Band 2400-2480 MHz 2400-2480 MHz RF Bandwidth 26 MHz 26 MHz Duplex Method Time Division Duplex Time Division Duplex Multiple Access GBT-CDMA GBT-CDMA Technique Number of Transmitter 2.sup.31 - 1 Chip-Sequences TX Data Rate: Traffic 384, 144, 128, 64, 32 384, 144, 128, 64, 32 Signalling/APC Kb/s Kb/s Forward Error Coding Rate - 1/2 Constraint Rate - 1/2 Constraint Length - 7 Length - 7 Convolutional Code Convolutional Code Interleaver 5 ms 5 ms Control Frame Length 500 .mu.sec 500 .mu.sec Data Modulation BPSK BPSK Spreading Technique Direct Sequence Direct Sequence Sequence Length 6,930,000 chips 6,930,000 chips Chip Rate 38.4 Mchips/s 38.4 Mchips/s Processing Gain 12 dB 12 dB Transmitter power 100 mW 100 mW (max) Service Range (free 0.8 km 0.8 km space) Number of Antenna omni omni Capacity 2 simultaneous users

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L14: Entry 1 of 5 File: USPT Nov 27, 2001

DOCUMENT-IDENTIFIER: US 6324164 B1
TITLE: Asynchronous transfer mode (A.T.M.) protocol adapter for a high speed cell switching system

Application Filing Date (1):
19980204

Brief Summary Text (7):
Prior art document "A Highly Modular Packet Switch for Gb/s Rates" by W. E. Denzel, A. P. J. Engbersen, I. Illiadis, G. Karlsson in XIV International Switching Symposium, October 1992, Vol. 2, page A8.3 ff. relates to a high-speed switching system.

Detailed Description Text (4):
On the other side, the retrieve section of switching module 401 comprises a set of sixteen Off-Chip-Drivers (OCD) drivers 11-0 to 11-15 which are used for interfacing the sixteen output ports of the switching module. The OCD drivers receive the data from sixteen routers 3-0 to 3-15 via an associated set of sixteen boundary latches 102-0 to 102-15 (used for timing considerations) so that each router 3-i can retrieve any data located within the 128 locations that are available into Cell Storage 1, and can transport them via a corresponding OCD Driver 11-i towards the appropriate destination output port I.

Detailed Description Text (7):
Additionally, EVEN bus 104 is connected to a first input bus of a MUX multiplexor 106 receiving at a second input the free addresses from FAQ 5 via bus 91. The output of MUX 106 is connected to a boundary latch 108, the output of which being connected to the inputs of a set of eight Off Chip Drivers (OCD) 40-0 to 40-7 and to a shadow latch 110. OCD Drivers 40-0 to 40-7 have outputs which are respectively connected to form an 8-bit bus 510 (formed of the eight outputs 510-0 to 510-7), also connected to the output of corresponding RCVR receivers 44-0 to 44-7. The outputs of RCVR receivers 44-0 to 44-7 are connected to a redundancy latch 180, which output is connected to one input bus of a MUX multiplexor 112, the second input of which receives the contents of shadow latch 110. MUX multiplexor 112 has an output that is connected to a pipeline Register 114 in order to load the data there through conveyed into the appropriate NSA Registers 22-0 to 22-7 as will be described hereinafter.

Detailed Description Text (8):
Similarly, ODD bus 105 is connected to a first input bus of a MUX multiplexor 107 receiving at a second input the free addresses from FAQ 5 via bus 92. The output of MUX 106 is connected to a boundary latch 109, the output of which being connected to the inputs of a set of eight Off Chip Drivers (OCD) 41-0 to 41-7 and to a shadow latch 111. OCD Drivers 41-0 to 41-7 have their outputs 509-0 to 509-7 which are respectively assembled in order to form an 8-bit bus 509, also connected to the inputs of eight RCVR receivers 45-0 to 45-7. The outputs of RCVR which output is connected to one input bus of a MUX multiplexor latch 111. MUX multiplexor 113 has an output that is connected to a pipeline Register 115 so that the addresses can be made available to the appropriate NSA Registers 23-0 to 23-7 as will be described

hereinafter.

Detailed Description Text (11):
NRA registers 28-0 to 28-7 are connected to receive the output of a MUX multiplexor circuit 26 which has a first and second input that respectively receives the contents of a shadow latch 34 and a boundary latch 80. Similarly, NRA registers 29-0 to 29-7 are connected to receive the output of a MUX multiplexor circuit 27 which has a first and second input that respectively receives the contents of a shadow latch 35 and a boundary latch 81. The output of latch 30 is connected to the input bus of shadow latch 34 and also to the inputs of a set of eight Off-Chip-Drivers (OCD) 42-0 to 42-7, which outputs 520-0 to 520-7 are assembled in order to form a bus 520 which also connected to the inputs of a set of eight RCV Receivers 46-0 to 46-7. Similarly, the output of latch 31 is connected to the input bus of shadow latch 35 and also to the inputs of a set of eight Off-Chip-Drivers (OCD) 43-0 to 43-7, which outputs 521-0 to 521-7, forming a bus 521, are connected to corresponding inputs of a set of eight RCVR Receivers 47-0 to 47-7. The outputs of RCVR Receivers 46-0 to 46-7 are connected to the input bus of latch 80, and the outputs of RCVR Receivers 47-0 to 47-7 are connected to the input bus of latch 81.

Detailed Description Text (115):
In the preferred embodiment of the invention, the Routing Control Devices are located within the switch core 450. This substantially enhances the possibilities of the switch since there becomes very simple to update the different contents of the multiple Control Routing Tables. Additionally, this presents the advantage of the possibility of using slower, cheaper and larger memory than that used for embodying Multicast table 6 which must be very rapid since it might occur that the latter is continuously in operation during one cell cycle). Further, the possibility of providing larger storage (also resulting from the fact that this storage may be located outside the chip of the switching module) for embodying Control Routing Tables permits to increase the number of routing SRH labels.

Detailed Description Text (118):
There is shown the switch core 1130 taking the form of one physical apparatus, which includes the switch structure 450, generally embodied under the form of a card comprising at least the four switching elementary modules, each module being an electronic chip. The two Routing control devices 1001-i and 1010-i that are associated to a same port i are embodied into a same physical chip 1110-i that is associated to a corresponding storage 1120-i that contains the two Routing Control Tables 1002-i and 1020-i described above in reference with FIG. 9. It therefore appears that switch structure 450 and the sixteen associated modules 1110 and 1120 are advantageously located in the same physical package, while the different SCAL elements are distributed in the different physical area of the industrial premisses where line attachment needs appear to be.

Detailed Description Text (120):
Further, it could be possible to use the teaching of document "Single-chip 4.times.500 Mbaud CMOS Transceiver" from A, Wilmer et al, in IEEE ISSCC96, Session 7, ATM/SOMET/PAPER FA 7.7. Published on Feb. 9th, 1996 for providing the possibilty of embodying the 1.6 Gigabit/s communication links 1400, 2400, 3400 and 4400 which is incorporated by simple reference. This document shows the possibility of using the so called 8B/10B. During idle periods that are marked by a flag, fill packets of data are transmitted, which start with a non-data Comma character. The Comma marks both byte and cell boundaries on the serial link. Therefore, synchronization at the byte and packet level can be provided and the 1, 6 Gigabit/s communication link may be embodied by means of an unique set of four optical cables, either coax or opticals. The reduction of the number of cables is substantial since, without this feature, at least five or six opticals lines would be necessary for embodying the 1.6 Gigabit/s communication link.

Detailed Description Text (124):

Now it will be described the general procedure that is used for creating and updating the Routing Control tables 1002-i and 1020-i which are located in the same chip. The procedure is illustrated in FIG. 12.

Detailed Description Text (147):
The processing of the second routing label proceeds then with the update of all the other tables 1002 and 1020. It should be noticed that the skilled man may advantageously loop the steps 1320 and 1330 in order to directly update the table 1002-i, before initiating the update process of table 1020-i. However such details of implementation will depend of the particular context and processor being actually used.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L14: Entry 2 of 5 File: USPT Mar 20, 2001

DOCUMENT-IDENTIFIER: US 6205543 B1
TITLE: Efficient handling of a large register file for context switching

Application Filing Date (1):
19981203

Brief Summary Text (9):
Another disadvantage of a large register file relates to the handling of registers during context switching of a multiprogrammed processor. A multiprogrammed processor is executable with several processes sharing the processing units concurrently. In any given clock cycle, only a single process has instructions executing on the processing units. The multiple processes execute concurrently by timesharing both the processing units and the memory, including the register file. When the context of the processor switches, the internal state of the processor, including all state information for an executing first process, is saved into a memory, and then state information for a saved second process is restored into an executing state. A processor with a large register file incurs a large overhead during context switching since the values for the first process that are held in a large number of registers are shifted from the register file to a context storage, followed by shifting of values for the second process from a context storage to the register file. The overhead of context switching reduces the time during which the processor executes instructions, reducing the efficiency of the processor.

Drawing Description Text (3):
FIG. 1 is a schematic block diagram illustrating a single integrated circuit chip implementation of a processor in accordance with ~~an embodiment of the present~~ invention.

Detailed Description Text (3):
~~Referring to FIG.~~ 1, a schematic block diagram illustrates a single integrated circuit chip implementation of a processor 100 that includes a memory interface 102, a geometry decompressor 104, two media processing units ~~110 and 112~~, a shared data cache 106, and several interface controllers. The interface controllers support an interactive graphics environment with real-time constraints by integrating fundamental components of memory, graphics, and input/output bridge functionality on a single die. The components are mutually linked and closely linked to the processor core with high bandwidth, low-latency communication channels to manage multiple high-bandwidth data streams efficiently and with a low response time. The interface controllers include a an UltraPort Architecture Interconnect (UPA) controller 116 and a peripheral component interconnect (PCI) controller 120. The illustrative memory interface 102 is a direct Rambus dynamic RAM (DRDRAM) controller. The shared data cache 106 is a dual-ported storage that is shared among the media processing units 110 and 112 with one port allocated to each media processing unit. The data cache 106 is four-way set associative, follows a write-back protocol, and supports hits in the fill buffer (not shown). The data cache 106 allows fast data sharing and eliminates the need for a complex, error-prone cache coherency protocol between the media processing units 110 and 112.

Detailed Description Text (4):

The UPA controller 116 is a custom interface that attains a suitable balance between high-performance computational and graphic subsystems. The UPA is a cache-coherent, processor-memory interconnect. The UPA attains several advantageous characteristics including a scaleable bandwidth through support of multiple bused interconnects for data and addresses, packets that are switched for improved bus utilization, higher bandwidth, and precise interrupt processing. The UPA performs low latency memory accesses with high throughput paths to memory. The UPA includes a buffered cross-bar memory interface for increased bandwidth and improved scaleability. The UPA supports high-performance graphics with two-cycle single-word writes on the 64-bit UPA interconnect. The UPA interconnect architecture utilizes point-to-point packet switched messages from a centralized system controller to maintain cache coherence. Packet switching improves bus bandwidth utilization by removing the latencies commonly associated with transaction-based designs.

Detailed Description Text (6):
Two media processing units 110 and 112 are included in a single integrated circuit chip to support an execution environment exploiting thread level parallelism in which two independent threads can execute simultaneously. The threads may arise from any sources such as the same application, different applications, the operating system, or the runtime environment. Parallelism is exploited at the thread level since parallelism is rare beyond four, or even two, instructions per cycle in general purpose code. For example, the illustrative processor 100 is an eight-wide machine with eight execution units for executing instructions. A typical "general-purpose" processing code has an instruction level parallelism of about two so that, on average, most (about six) of the eight execution units would be idle at any time. The illustrative processor 100 employs thread level parallelism and operates on two independent threads, possibly attaining twice the performance of a processor having the same resources and clock rate but utilizing traditional non-thread parallelism.

Detailed Description Text (8):
Although the processor 100 shown in FIG. 1 includes two processing units on an integrated circuit chip, the architecture is highly scaleable so that one to several closely-coupled processors may be formed in a message-based coherent architecture and resident on the same die to process multiple threads of execution. Thus, in the processor 100, a limitation on the number of processors formed on a single die thus arises from capacity constraints of integrated circuit technology rather than from architectural constraints relating to the interactions and interconnections between processors.

Detailed Description Text (13):
Each media processing unit 110 and 112 includes a split register file 216, a single logical register file including 128 thirty-two bit registers. The split register file 216 is split into a plurality of register file segments 224 to form a multi-ported structure that is replicated to reduce the integrated circuit die area and to reduce access time. A separate register file segment 224 is allocated to each of the media functional units 220 and the general functional unit 222. In the illustrative embodiment, each register file segment 224 has 128 32-bit registers. The first 96 registers (0-95) in the register file segment 224 are global registers. All functional units can write to the 96 global registers. The global registers are coherent across all functional units (MFU and GFU) so that any write operation to a global register by any functional unit is broadcast to all register file segments 224. Registers 96-127 in the register file segments 224 are local registers. Local registers allocated to a functional unit are not accessible or "visible" to other functional units.

Detailed Description Text (15):
Instructions are executed in-order in the processor 100 but loads can finish out-of-order with respect to other instructions and with respect to other loads, allowing loads to be moved up in the instruction stream so that data can be

streamed from main memory. The execution model eliminates the usage and overhead resources of an instruction window, reservation stations, a re-order buffer, or other blocks for handling instruction ordering. Elimination of the instruction ordering structures and overhead resources is highly advantageous since the eliminated blocks typically consume a large portion of an integrated circuit die. For example, the eliminated blocks consume about 30% of the die area of a Pentium II processor.

Detailed Description Text (20):
The illustrative processor 100 has a rendering rate of over fifty million triangles per second without accounting for operating system overhead. Therefore, data feeding specifications of the processor 100 are far beyond the capabilities of cost-effective memory systems. Sufficient data bandwidth is achieved by rendering of compressed geometry using the geometry decompressor 104, an on-chip real-time geometry decompression engine. Data geometry is stored in main memory in a compressed format. At render time, the data geometry is fetched and decompressed in real-time on the integrated circuit of the processor 100. The geometry decompressor 104 advantageously saves memory space and memory transfer bandwidth. The compressed geometry uses an optimized generalized mesh structure that explicitly calls out most shared vertices between triangles, allowing the processor 100 to transform and light most vertices only once. In a typical compressed mesh, the triangle throughput of the transform-and-light stage is increased by a factor of four or more over the throughput for isolated triangles. For example, during processing of triangles, multiple vertices are operated upon in parallel so that the utilization rate of resources is high, achieving effective spatial software pipelining. Thus operations are overlapped in time by operating on several vertices simultaneously, rather than overlapping several loop iterations in time. For other types of applications with high instruction level parallelism, high trip count loops are software-pipelined so that most media functional units 220 are fully utilized.

Detailed Description Text (25):
The new media data that is operated upon by the processor 100 is typically heavily compressed. Data transfers are communicated in a compressed format from main memory and input/output devices to pins of the processor 100, subsequently decompressed on the integrated circuit holding the processor 100, and passed to the split register file 216.

Detailed Description Text (26):
Splitting the register file into multiple segments in the split register file 216 in combination with the character of data accesses in which multiple bytes are transferred to the plurality of execution units concurrently, results in a high utilization rate of the data supplied to the integrated circuit chip and effectively leads to a much higher data bandwidth than is supported on general-purpose processors. The highest data bandwidth requirement is therefore not between the input/output pins and the central processing units, but is rather between the decompressed data source and the remainder of the processor. For graphics processing, the highest data bandwidth requirement is between the geometry decompressor 104 and the split register file 216. For video decompression, the highest data bandwidth requirement is internal to the split register file 216. Data transfers between the geometry decompressor 104 and the split register file 216 and data transfers between various registers of the split register file 216 can be wide and run at processor speed, advantageously delivering a large bandwidth.

CLAIMS:

20. A method of switching context in a processor that includes an executive storage for holding operand data operated upon by instructions executing on the processor, the executive storage being divided into a plurality of storage groups containing one or more storage elements, the method comprising:

utilizing a dirty bit storage including a plurality of storage bits corresponding to a plurality of respective storage groups in the executive storage;

receiving a destination address field of the executing instructions;

responsive to an executed instruction, classifying a destination access as a targeted storage group according to information in the destination address field of the executed instruction;

evaluating the classified destination based on whether the instruction updates the targeted storage group; and

responsive to a context switch, saving storage groups based on the evaluation of the classified destinations.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L14: Entry 3 of 5 File: USPT Feb 27, 2001

DOCUMENT-IDENTIFIER: US 6195739 B1
TITLE: Method and apparatus for passing data among processor complex stages of a pipelined processing engine

Application Filing Date (1):
19980629

Brief Summary Text (9):
A router is an intermediate station that implements network services such as route processing, path determination and path switching functions. The route processing function determines the type of routing needed for a packet, whereas the path switching function allows a router to accept a frame on one interface and forward it on a second interface. The path determination, or forwarding decision, function selects the most appropriate interface for forwarding the frame. A switch is also an intermediate station that provides the basic functions of a bridge including filtering of data traffic by medium access control (MAC) address, "learning" of a MAC address based upon a source MAC address of a frame and forwarding of the frame based upon a destination MAC address. Modern switches further provide the path switching and forwarding decision capabilities of a router. Each station includes high-speed media interfaces for a wide range of communication links and subnetworks.

Brief Summary Text (16):
Increases in the frame/packet transfer speed of an intermediate station are typically achieved through hardware enhancements for implementing well-defined algorithms, such as bridging, switching and routing algorithms associated with the predefined protocols. Hardware implementation of such an algorithm is typically faster than software because operations can execute in parallel more efficiently. In contrast, software implementation of the algorithm on a general-purpose processor generally performs the tasks sequentially because there is only one execution path. Parallel processing of conventional data communications algorithms is not easily implemented with such a processor, so hardware processing engines are typically developed and implemented in application specific integrated circuits (ASIC) to perform various tasks of an operation at the same time. These ASIC solutions, which are generally registers and combinational logic configured as sequential logic circuits or state machines, distinguish themselves by speed and the incorporation of additional requirements beyond those of the basic algorithm functions. However, the development process for such an engine is time consuming and expensive and, if the requirements change, inefficient since a typical solution to a changing requirement is to develop a new ASIC.

Detailed Description Text (3):
FIG. 2 is a schematic block diagram of intermediate station 200 which, in the illustrative embodiment, is preferably a network switch. The switch generally performs layer 2 processing functions, such as "cut-through" operations wherein an entire frame does not have to be stored before transfer to a destination; in addition, switch 200 may implement layer 3 forwarding operations. It should be noted, however, that the intermediate station may also be configured as a router to perform layer 3 route processing. A feature of the inventive architecture described

herein is the ability to program the station for execution of either layer 2 and layer 3 operations. To that end, operation of the switch will be described with respect to IP switching of packets, although the switch may be programmed for other applications, such as data encryption.

Detailed Description Text (6):
The arrayed processing engine 300 is coupled to a memory partitioned into a plurality of external memory (Ext Mem) resources 280. A buffer and queuing unit (BQU) 210 is connected to a packet memory 220 for storing packets and a queue memory 230 for storing network layer headers of the packets on data structures, such as linked lists, organized as queues 235. The BQU 210 further comprises data interface circuitry for interconnecting the processing engine with a plurality of line cards 240 via a selector circuit 250. The line cards 240 may comprise OC12, OC48 and Fast Ethernet (FE) ports, each of which includes conventional interface circuitry that incorporates the signal, electrical and mechanical characteristics, and interchange circuits, needed to interface with the physical media and protocols running over that media. A typical configuration of the switch may include many (e.g., thousands) input/output channels on these interfaces, each of which is associated with at least one queue 235 in the queue memory 230. The processing engine 300 generally functions as a switching processor that modifies packets and/or headers in sequence as the BQU 210 implements queuing operations.

Detailed Description Text (7):
A route processor (RP) 260 executes conventional routing protocols for communication directly with the processing engine 300. The routing protocols generally comprise topological information exchanges between intermediate stations to determine optimal paths through the network based on, e.g., destination IP addresses. These protocols provide information used by the RP 260 to create and maintain routing tables. The tables are loaded into the external partitioned memories 280 as forwarding information base (FIB) tables used by the processing engine to perform forwarding operations. When processing a header in accordance with IP switching, the engine 300 determines where to send the packet by indexing into the FIB using an IP address of the header. Execution of the forwarding operations results in destination media access control (MAC) addresses of the headers being rewritten by the processing engine to identify output ports for the packets.

Detailed Description Text (11):
The CPU 410 is preferably a small processor core having a dense structure which enables implementation of similar cores on an application specific integrated circuit (ASIC). In the illustrative embodiment described herein, the CPU is a 32-bit, 100 MHz Advanced RISC Machine (ARM) 7TDI core capable of executing 16-bit or 32-bit instructions; however, it will be apparent to those skilled in the art that other CPU cores may be advantageously used with the processor complex architecture described herein. The ARM CPU includes an arithmetic logic unit (ALU), internal registers for storing information processed by the ALU, and an instruction fetch and decode unit that decodes instructions fetched from the instruction memory. The instructions are generally vertical assembly language code manifested in the form of conventional reduced instruction set computer (RISC) instructions.

Detailed Description Text (32):
At the end of the phase, context for each processor complex is switched to the alternate upstream and downstream context memories, and a new phase begins. Notably, all CPUs 410 of the engine 300 are synchronized to either phase A or B; the phase further determines from which context memory (CMA or CMB) the CPU fetches data. Thus, the downstream context for one CPU becomes the upstream context for the next CPU in the new phase, allowing seamless transfer of context data from CPU to CPU in a serial manner.

Detailed Description Text (34):

In summary, the context passing technique described herein enhances the speed of data execution in a pipelined processing engine by substantially reducing the latency involved with passing the data among stages of the engine. Note that transient context data entering the engine is dispatched to a processor complex stage of a pipeline for processing by a CPU prior to serially passing the data to a downstream processor complex stage. The processor complex architecture described herein facilitates the passing of transient data from an upstream context memory to a corresponding downstream memory as the CPU processes the data. Moreover, the data mover may be programmed to inconspicuously move any type of context data, such as network layer headers in the case of packet switching operations or the contents of entire data frames in the case of encryption operations, from the context memories (ping-pong buffers) during such CPU processing. Thus, the invention transforms an otherwise serial data processing/passing procedure to a parallel process.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L14: Entry 5 of 5 File: USPT Apr 14, 1992

DOCUMENT-IDENTIFIER: US 5105424 A
TITLE: Inter-computer message routing system with each computer having separate routinng automata for each dimension of the network

Application Filing Date (1):
19880602

Brief Summary Text (5):
Dally, William and Charles L. Seitz, "The Torus Routing Chip," Distributed Computing, Vol. 1, No. 4, pp 187-196, Springer-Verlag, October 1986.

Brief Summary Text (19):
The Torus Routing Chip (TRC) designed at Caltech in 1985 [Dally & Seitz 86] used unidirectional channels between the nodes 10 connected in a torus as shown in FIG. 3. This is also the subject of a patent application entitled Torus Routing Chip by Charles L. Seitz and William J. Dally, Ser. No. 944,842, Filed Dec. 19, 1986, and assigned to the common assignee of this application, the teachings of which are incorporated herein by reference. As depicted in FIG. 3, the torus is shown folded in its projection onto a common plane in order to keep all channels the same length. Deadlock (a major consideration in multicomputers) was avoided by using the concept of virtual channels, by which a packet injected into a network travels along a spiral of virtual channels, thus avoiding cyclic dependencies and the possibility of deadlock. The TRC was self-timed to avoid the problems associated with delivering a global clock to a large network. There were a total of 5 channels to deal with, i.e., channels to and from the node and 2 virtual channels each in x and y. Thus, the heart of the TRC involved a 5.times.5 crossbar switch. Although the initial version had a slow critical path, the revised version was expected to operate at 20MHz, with a latency from input to output of 50ns. Since each channel had 8 data lines, the TRC achieved a data rate of 20MB/s. Each packet is made up of a header, consisting of 2 bytes containing the relative x and y address of the destination, any number of non-zero data bytes, and a zero data byte signifying a "tail" or end of the packet. Upon entering the router, each packet has the address in its header decremented and tested for zero and is then passed out through the proper output channel. The connection stays open for the rest of the message and closes after passage of the tail (wormhole routing). If the desired output channel is unavailable, the message is blocked until the channel becomes available.

Brief Summary Text (20):
In the winter and spring of 1986, concurrently with the developments described above, groups of students in the "VLSI Design Laboratory" project course, under the direction of Dr. Charles Seitz of Caltech, were put to work designing different parts of the "Mosaic C" element. This single-chip node of a message-passing multicomputer was to contain a 16-bit central processing unit (CPU), several KBytes of on-chip dynamic random access memory (dRAM), and routing circuitry for communication with other chips. Each chip would form a complete node in a so-called fine-grain concurrent computer.

Brief Summary Text (21):
After looking at a few possible implementations, including the TRC described above,

the group working on the routing section decided that a simple, bidirectional 2-D mesh should be used. A mesh had the advantage of keeping the length of wires between chips down to less than one inch, which would allow the use of a synchronous protocol, since clock skew as a function of wire length could be made very small between chips. A mesh would also allow the channels at the edge of the array to be reserved for communications with the outside world. The group also decided to use a bit-serial protocol for packets, both to minimize the number of pins on each chip and to minimize the number of connections needed between them; but, to organize the packets into flits sufficiently large that all of the routing information could be contained in the first flit. As in the TRC, the first Mosaic C router as specified by this group was to use virtual channels to avoid the possibility of deadlock. Each packet consisted of a 20-bit header with the relative x and y addresses of the destination and an arbitrary number of 20-bit flits consisting of a 16-bit data word and 4 control bits. The router also used wormhole routing with one of the control bits signifying a tail. Internally, flits were switched between input and output channels using a time multiplexed bus. The control circuitry was kept as simple as possible, and as a result, did not know how to forward a packet by itself. Each time the header of a packet came in, the CPU would be interrupted (using a dual-context processor for fast interrupt handling) to determine which output channel the packet should be connected to. This approach resulted in a latency of several micro-seconds per step in path formation, but allowed a lot of flexibility in routing under software control. Acknowledgement packets would automatically be sent and received between chips using the same channels to announce the availability of buffers. With a 20MHz system clock (anticipated for 2 micrometer CMOS technology), the bandwidth was expected to be about 2MB/s on each channel. This initial attempt at a routing circuit for incorporation into the Mosaic C chip was never reduced to a layout. After due consideration, it became obvious that it would consume a large amount of silicon area (on the chip) only to achieve fairly dismal performance.

Brief Summary Text (22):
Wherefore, it is an object of the present invention to provide a new method for routing message packets in a message-passing, multicomputer system which will allow the routing processor to provide good performance with a minimum amount of silicon area on the chip consumed thereby.

Brief Summary Text (24):
It is still another object of the present invention to provide a multifunction node chip for use in fine grain message-passing, multicomputer systems incorporating a router for routing message packets in a manner to provide good performance with a minimum amount of silicon area on the chip consumed thereby.

Brief Summary Text (27):
The foregoing objects have been achieved in a fine-grain, message-passing, multicomputer, concurrent computing system wherein there are a plurality of computing nodes each including bus means for interconnecting the components of the chip; read only memory (ROM) operably connected to the bus means; random access memory (RAM) operably connected to the bus means; central processing unit (CPU) means operably connected to the bus means for executing instructions contained in the ROM and RAM; and packet interface (PI) means operably connected to the bus means for encoding headers on message packets being transmitted by the CPU means of one chip to another chip and for transferring the message packets to and from the RAM, by the improved method of routing the message packets between the nodes comprising the steps of, providing a routing automaton at each node and interconnecting the routing automata to define communications paths interconnecting the nodes along which the message packets can be routed; and at each routing automaton, receiving a message packet including routing directions comprising the header at an input thereof; reading the header; directing the message packet to one of two outputs thereof as a function of the routing directions contained in the header; and, updating the header to reflect the passage of the message packet

through the routing automaton.

Drawing Description Text (4):
FIG. 3 is a simplified drawing depicting a prior art torus routing chip interconnection scheme.

Drawing Description Text (7):
FIG. 6 is a simplified functional block diagram of a fine-grain computer chip according to the present invention.

Drawing Description Text (12):
FIG. 11 is a simplified block diagram of the internal structure of a routing automaton according to the present invention in an embodiment thereof intended for use in a torus routing chip system of the type shown in FIG. 3.

Drawing Description Text (14):
FIG. 13 is a drawing corresponding to the embodiment of FIG. 10 and depicting the elements thereof in their connected sequence as incorporated into a chip as laid out, built and tested by the applicant herein.

Detailed Description Text (3):
The second major deviation from the prior art was the early rejection of the crossbar switch in favor of a new formulation which has been designated as "routing automata". It was realized early on that a crossbar switch, while very general, had the disadvantage of taking up space (on the chip) proportional to (nW).sup.2, where n is the number of inputs and outputs, and W is the number of bits being switched. With a fixed routing scheme being employed, the generality of a crossbar switch was not needed and it was highly desirable to devise a scheme in which the area consumed would only increase linearly with nW, or as close as possible thereto. This scaling would make it easier to modify the router for more dimensions or wider flits without involving major layout changes, would decrease the path length in the switch and hence its speed, and would, hopefully, decrease the overall area for designs with wide flits and a large number of dimensions.

Detailed Description Text (4):
As depicted in FIG. 4, each of the automata 12 is responsible for switching the packet streams for one dimension of the overall router. An automaton's input, generally indicated as 14, consists of streams from the + and - directions as well as from the previous dimension. Its output, generally indicated as 16, consists of streams to the + and - directions as well as to the next dimension. For n dimensions, n of the automata 12 are strung together in series as depicted in FIG. 5; and, if properly constructed, their size, i.e. the area consumed on the chip, increases roughly linearly with increased width of the flits, for a net increase in area proportional to nW, as desired. Both synchronous and asynchronous (i.e. self-timed) versions of automata according to the present invention will be described shortly. The self-timed version is intended to have each of its components highly modular so that they could be used not only to implement mesh routing, as in the Mosaic C chip, but could also be fit together to implement unidirectional routers, routers for hypercubes, or many other structures, limited only by the desires of the designer. The basic components include a FIFO for "glue" between stages and buffering, a switch used both to divide and merge data streams, a decrementer for adjusting the relative address of the destination as the packet passes through, and control structures for all of the foregoing. Before continuing with specifics of the automata, however, the novel prefix encoding scheme of the present invention as employed therein will be addressed first in further detail.

Detailed Description Text (6):
In the example, the message TMMMM.3+12+ consists of the header (.3+12+), the "payload" (MMMM), which could be of any length, and the tail flit (T). When this message is injected by the SOURCE node 10, the route to the DESTINATION node 10 is

to go six nodes in the + direction from the SOURCE node 10 (i.e. 12 radix 4=1(4)+2 (1)=6) and then three nodes in the + direction from NODE 6. Originally, the packet enters the network and takes the + direction in the first (i.e. x) dimension. The + flit is stripped off and the new leading flit (i.e. 2) is decremented while passing through each node until it reaches 0. During the decrement to 0 (in this example in NODE 2), the following flit is examined to see if it is a digit. In this case it is, so the leading flit becomes 0. In NODE 3, the second flit needs to be decremented to 0; but, since it is not followed by a digit, it becomes a "leading zero", indicated by the "." designator. The leading flit becomes a 3 (representing the 4 additional nodes to proceed in the same direction) and the leading zero indicator is placed in the following flit. In NODE 6, the leading flit is once again in the position of decrementing to 0. This time, however, the next flit is the leading zero designator. As a consequence, the third flit is used to switch the path of the packet to the + direction in the second (i.e. y) dimension from NODE 6. As with the initial + indicator, the + indicator and two leading zero (i.e. "..") flits are stripped from the header. The same process is continued until the relative address of the header is decremented to 0 once again. At this point, the packet has run out of dimensions to traverse, so it is passed into the receiving (i.e. DESTINATION) node 10. The tail (T) which makes up the end of the packet closes all of the channel connections as it passes through the nodes 10, and is finally stripped off at the DESTINATION node 10 along with the remaining header flits to leave only the data flits as indicated in FIG. 7. As can be appreciated from this example, the encoding scheme of the present invention allows the use of small flits to represent large offsets (compared with prior art x,y final address designations) while allowing decisions to be made based on only two flits of the header at once, which helps minimize the latency of forwarding through each node. The simple decision involved also allow a simple control logic to be employed.

Detailed Description Text (7):
Turning once again to the routing automata of the present invention, as previously mentioned, the reduction of the routing circuitry to simple automata that control the switching through only one dimension greatly simplifies the modification and expansion of a complete router. An individual automaton is also much easier to design and lay out due to the reduced number of inputs and outputs, and independence from the routing occurring in other dimension. As pointed out above, the basic one dimensional (1-D) automaton 12 of FIG. 4 has three inputs 14 for the receipt of packets travelling, respectively, (1) in the + direction, (2) in the - direction and (3) from the previous dimension. Simple finite state machines can then process the input streams, decide on a switch configuration that allows the largest number of packets to be forwarded, and then connect the streams (1) to the + direction, (2) to the - direction and (3) to the next dimension at the outputs 16.

Detailed Description Text (11):
The invention of the routing automata and its associated method of operation in routing packets provided the opportunity of constructing a novel chip for the Mosaic C application as well; that is, the present invention includes a novel chip for containing each node in a fine-grain, message-passing, multicomputer, concurrent computing system wherein a plurality of computing nodes are each contained on a single chip. This "Mosaic chip" 22 is shown in functional block diagram form in FIG. 6. The Mosaic chip 22 is a complete node for a fine-grain concurrent computer. It contains all of the necessary elements including a 16-bit CPU 24 several KBytes of ROM 26 and dRAM 28, as well as routing circuitry comprising a packet interface (PI) 30 and router 32 for communicating with neighboring nodes in a mesh. All of these elements are tied to a common bus 36. Since the router had to fit on a chip along with a processor and memory, the design had to be simple and compact--the automata-based routing scheme of the present invention as described hereinbefore provided such a capability. The PI 30 takes care of encoding the packet header and transferring packet data to and from memory. A simple cycle-stealing form of Direct Memory Access (DMA) is preferred to keep up

with the high data rates supported by the router. In the preferred embodiment, the PI 30 also contains some memory mapped locations that are used to specify the relative x and y addresses of the destination node, and an interrupt control register.

Detailed Description Text (13):
As will be described in greater detail hereinafter, the Mosaic router 34 communicates with other nodes using a 3-bit wide flit (2 data, 1 control) with an acknowledge wire in the reverse direction to control movement between stages of the preferred pipelined design and prevent overwriting from one stage to the next. Any time an acknowledge is present, flits are allowed to progress through the pipeline. The flit can also be made wider to include more data bits. As presently contemplated, the first "production" Mosaic chip will employ a 5-bit flit.

Detailed Description Text (15):
In an attempt to minimize the overhead of extending the width of a flit, the Mosaic router as built and tested employed an approach of constructing the data path out of 1-bit wide slices, with the +, -, and N paths for each bit being placed immediately next to one another. This preferred approach allows the same switching elements to be used no matter how wide the flit is. On the negative side, it also means that the control signals for all three data paths had to be propagated through all of the elements, and this led to a somewhat larger overhead in wiring than is necessary. It also complicated the layout of the control circuitry since it had to fit in an effectively smaller pitch. The data path is made up of a number of elements as shown in FIG. 13 (which corresponds to the embodiment of FIG. 10 as actually laid out and implemented on a chip). Each element, of course, comprises three portions (for the +, - and N paths) as indicated by the dashed lines dividing them. When connected sequentially in the order shown, these elements form a complete data path for a Mosaic routing automaton. The inputs are to an input latch 38 followed by an input shift register 40 as required to properly interface with the preceding stage. This is followed by the zero and tail detection logic 42. Next follows the decrementer and leading zero generator logic 44. The stream switching element 46 follows next. As indicated by the dashed arrows, the stream switching element portions operate in the same manner as the three decision elements 18 of FIG. 10; that is, the two outer switching element portions switch between straight ahead and towards the center path while the center switching element portion can switch between straight ahead and either of the two outer paths. The properly switched packet paths from the stream switching element 46 then proceeds to an output latch 48 and an output buffer/disabler 50 as required to properly interface the asynchronous automata to the next node without destructive interference, as described earlier herein.

Detailed Description Text (18):
Much of what was learned from the design of the Mosaic synchronous router can be applied to an asynchronous routing automaton. An asynchronous router can be used in physically larger systems, such as second-generation multicomputers, in which the interconnections are not limited to being very short wires. The mesh routing chip (MRC) now to be described is designed to meet the specifications for these second-generation multicomputers. These routing automata are intended to be a separate chip, similar to the Torus Routing Chip mentioned earlier herein, as opposed to being part of an integrated "total node" chip such as the Mosaic chip described above. As in the Mosaic router, the 2-D MRC has 5 bidirectional channels, with channels in the +x, -x, +y, -y directions and a channel connecting it to the packet interface. Data is represented on each of the channels using 9-bit wide flits (1 tail, 8 data), where the first bit is the tail bit.

Detailed Description Text (19):
For a 2-D router, the first two flits form the header. In each header flit, a relative offset of six bits allows for up to sixty four nodes along a single dimension, which should be sufficient for any second generation machine with large

nodes. The seventh data bit in a header flit is reserved for the future addition of broadcast support and the eighth bit is the sign. A 9-bit flit together with the asynchronous request and acknowledge signals for each channel requires a total of eleven pins. Five directional channels (+x, -x, +y, -y, and the node) then require 110 pins, and the constructed version of the chip was placed in a 132 pin PGA package. The remaining pins are used for a reset and for multiple Vdd and GND pins to minimize noise.

Detailed Description Text (22):
Looking back at the three dimensional automata series shown in FIG. 5, it can be seen that data flows in only one direction within each automaton 12, and the different paths are independent (except for merge operations). Thus, it is an easy transition to think of routing automata as being implemented using a pipeline structure, as in the MRC. As established earlier here, the transit time, from source to destination, of an unblocked packet in the synchronous case is given by, T.sub.n =T.sub.c (pD+L/W). For the asynchronous case, some of these terms are changed because the head of a packet advances at the fallthrough rate, which is less than the cycle time. The formula for network latency of the asynchronous case can be expressed as, T.sub.n =T.sub.f D+T.sub.c [L/W]; where T.sub.f is the fallthrough time for a node. For relatively short packets, D is comparable to L/W, so there is no strong motivation to reduce either T.sub.f or T.sub.c at the expense of the other. T.sub.f and T.sub.c can be expressed as, T.sub.c .apprxeq.2t.sub.p +t.sub.c and T.sub.f .apprxeq.t.sub.p +t.sub.f p; where, t.sub.p is the time required to drive the pads, p is the number of stages in the internal pipeline, and t.sub.f and t.sub.c are the average fallthrough and cycle times, respectively, for a single stage of the pipeline, as described above. A pad, and the external components connected to it, are relatively difficult for a VLSI chip to drive, so t.sub.p >>t.sub.c >t.sub.f. This means that the number of stages in an asynchronous pipeline can be increased without significantly increasing the overall delay. In the case of the MRC, increased pipelining has a significant advantage in that having more pipeline stages provides the network with more internal storage for packets, and consequently, helps prevent congestion of the network.

Detailed Description Text (27):
Finally, it should be noted that, internally, the automata of the present invention use 4-cycle signaling for flow control; but, to increase speed and conserve power, signals sent off chip must use a 2-cycle convention. A small amount of conversion must be done, therefore, before driving the pads. This conversion also adds a small amount of delay in the request/acknowledge path, which helps ensure that the data is valid by the time a request is received, even if the delays in the lines are slightly skewed. If the delays are skewed by a large amount, a simple lumped RC delay can be added to the request line external to the chip.

Detailed Description Text (28):
It is worthy of note to report that the first Asynchronous MRC chips were suubmitted in September 1987 for prototype fabrication in a 3 micrometer CMOS process. The fabricated and packaged chips were returned in November 1987. They functioned correctly at a speed of approximately 10 Mflits/s. Production chips' fabricated later in a 1.6 micrometer CMOS process operated at about 30 Mflits/s, about three times faster. Subsequent design refinements in layout and implementation have increased the potential speed of the MRC to 48 Mflits/s in 3 micrometer CMOS. This speed obtained by a prototype of FIFO and 2/4 cycle conversion circuitry submitted for fabrication in February 1988 and returned and tested in April 1988. In a 1.6 micrometer CMOS process, these designs are anticipated to operate at approximately 100 Mflits/s. The primary limitation at these speeds is the lead inductance of PGA packages. Improved packaging techniques should allow such chips to operate at 150 Mflits/s.

CLAIMS:

14. The system of claim 1 wherein each of said computers comprise an integrated circuit including the corresponding router, a processor, a memory, a packet interface connected to said router and a bus connecting said packet interface, said memory and said processor, wherein:

said processor comprises means for retrieving information data from said memory to be routed in said network in a message packet and for computing an initial header based upon destination instructions stored in said memory; and

said packet interface comprises means for forming a data stream comprising said information data and said initial header and transmitting said data stream to said router connected thereto as a message packet.

22. The system of claim 17 wherein each of said computers comprise an integrated circuit including the corresponding router, a processor, a memory, a packet interface connected to said router and a bus connecting said packet interface, said memory and said processor, wherein:

said processor comprises means for retrieving information data from said memory to be routed in said network in a message packet and for computing an initial header based upon destination instructions stored in said memory; and

said packet interface comprises means for forming a data stream comprising said information data and said initial header and transmitting said data stream to said router connected thereto as a message packet.

25. A computer node chip for use in an inter-computer message routing system wherein message packets are routed among a plurality of such computer node chips along communication paths between said computer node chips in an n-dimensional network of said communication paths, different groups of said communication paths comprising different ones of the n dimensions of said network, said message packets each comprising a header containing successive routing directions relative to successive computer node chips along a selected route in said network, said computer node chip comprising:

a router comprising n routing automata corresponding to said n dimensions, each of said n routing automata having plural message packet inputs and plural message packet outputs, at least some of said inputs and outputs being connected to respective communication paths of the corresponding one of said n dimensions, said n routing automata being connected together in cascade from a message packet output of one to a message packet input of the next one of said routing automata corresponding to a sequence of dimensions of the routing automata;

routing logic means disposed within each one of said routing automata, said routing logic means comprising means for reading the header of a message packet received from one of the inputs of said one routing automata, means for directing said message packet to one of said outputs of said one routing automata in accordance with the contents of said header, and means for modifying said header to reflect the passage of said message packet through said one routing automata, whereby each of said routing automata performs all message routing for the message packets traveling in a corresponding one of said dimensions;

a processor;

a memory; and

a packet interface connected to said router and a bus connecting said packet interface, said memory and said processor, wherein said processor comprises means for retrieving information data from said memory to be routed in said network in a message packet and for computing an initial header based upon destination

instructions stored in said memory, and said packet interface comprises means for forming a data stream comprising said information data and said initial header and transmitting said data stream to said router connected thereto as a message packet.

26. The chip of claim 25 wherein:

said means for directing said message packet to one of said outputs comprises plural decision element means each having one decision input and plural decision outputs for directing a message packet received at said decision input to one of said decision outputs in accordance with the contents of said header, and plural merge element means having plural merge inputs and a single merge output for directing message packets received at the plural merge inputs thereof to said merge output without interference between said message packets; and

within a single one of each routing automata at least some of said decision inputs are connected to respective message packet inputs, at least some of said merge inputs are connected to respective decision outputs and at least some of said merge outputs are connected to respective message packet outputs.

Previous Doc Next Doc Go to Doc#